EISTI Ecole Internationale des Sciences du Traitement de l'Information	**ING 1 : EXAMEN D'ALGORITHMIQUE PROCEDURALE**
Rédigé par : l'équipe pédagogique du module Algo. Procédurale	Ref : *ING1-ALG-PROC-EXAMEN*
A l'intention de : Etudiants des ING1	Créé le : 10/01/2013

Préambule

Cet examen dure 2h00. L'examen se fait sur feuilles. Aucun document n'est autorisé. L'ordinateur est interdit. On vous demande de mentionner dans votre copie le libellé de votre groupe.

1. Tableaux, tris et structures

On définit la structure suivante :

```
Type EleveStructure
        nom : Chaine
        prenom : Chaine
        notes : Tableau de Reel(10)
        moyenne : Reel
Fin Type

Type GroupeEleves Structure
        libelle : Chaine // libellé du groupe
        n : Entier // nombre d'élèves dans le groupe
        eleves : tableau de Eleve(40)
Fin Type
```

On dispose d'un groupe d'au plus 40 élèves. On suppose que pour chaque élève, on a déjà saisi les 10 notes qu'il a obtenues. Il s'agit maintenant de faire deux choses :

1. Pour chaque élève, il faut calculer le champ moyenne. On note **calculerMoyenne** cette procédure.
2. Dans le groupe, de classer le tableau d'élèves par ordre décroissant de moyenne sachant que pour un ensemble d'élèves ayant la même moyenne, on les classe par ordre alphabétique nom puis prénom. On note **trierGroupe** cette procédure.

Chaque note a un poids dans le calcul de la moyenne. On suppose que la moyenne générale est la moyenne pondérée des 10 notes avec ces coefficients.

- Expliquer pourquoi le prototype de méthode de calculerMoyenne est :
  Procédure calculerMoyenne(eleve : Eleve (E/S) , coeff : tableau de Reel(10) (E))
- Ecrire l'algorithme de cette procédure.
- Expliquer pourquoi le prototype de méthode de trierGroupe est :
  Procédure trierGroupe(groupe : GroupeEleves (E/S))
- Ecrire l'algorithme de cette procédure.
- Ecrire la procédure qui fait le tout.

2. Structures séquentielles

On considère des expressions numériques composées de valeurs numériques, des opérateurs binaires +, -, * et /, de parenthèses ouvrantes et de parenthèses fermantes. Cet exercice consiste à convertir une expression infixée en une expression postfixée.

Une expression infixée est une expression habituelle que vous utilisez depuis toujours avec les priorités d'opérateurs de la plus forte à la plus faible suivantes :

- les parenthèses ouvrantes et fermantes
- les opérateurs * et /
- les opérateurs + et –

Une expression postfixée est définie comme suit :

- un opérande est une expression postfixée,
- si x et y sont deux expressions postfixées et op un opérateur binaire alors **x y op** est une expression postfixée,
- il n'y a pas d'autre expression postfixée.

On explique ci-dessous un algorithme de conversion en langage naturel :

- En entrée, on utilise un flot de termes contenant l'expression infixée
- En intermédiaire, on utilise une pile p
- En sortie, on utilise une file s

On initialise p et s à vide.

On parcourt le flot d'entrée tant qu'il y a des termes dans le flot. Pour un terme **t** donné, on effectue l'une des opérations suivantes :

- le terme **t** est un opérande alors on l'enfile dans s,
- le terme **t** est une parenthèse ouvrante, on l'empile dans p,
- le terme **t** est une parenthèse fermante, on dépile p jusqu'à trouver la première parenthèse ouvrante. Chaque fois qu'un terme est dépilé, on l'enfile dans s sauf la parenthèse ouvrante.
- le terme **t** est un opérateur, tant que l'opérateur du haut de la pile a une priorité plus forte que celle du terme t alors on dépile et on enfile l'opérateur dépilé dans s. A la fin de ce dépilement, on empile **t** dans p.

A la fin de la lecture du flot, on vide la pile p dans la file s

Afin de mieux comprendre, on traite un exemple. On considère l'expression infixée **a + b * (c – (d * e + f))**. Dans ce qui suit, le haut de la pile p est à droite et la tête de la file s est à gauche.

Lecture dans le flot	Pile p	File s
Le terme a		a
Le terme +	+	a
Le terme b	+	a b
Le terme *	+ *	a b
Le terme (	+ * (	a b
Le terme c	+ * (	a b c
Le terme -	+ * (-	a b c

Lecture dans le flot	Pile p	File s
Le terme (	+ * (- (	a b c
Le terme d	+ * (- (	a b c d
Le terme *	+ * (- (*	a b c d
Le terme e	+ * (- (*	a b c d e
Le terme +	+ * (- (+	a b c d e *
Le terme f	+ * (- (+	a b c d e * f
Le terme)	+ * (-	a b c d e * f +
Le terme)	+ *	a b c d e * f + -

On vide la pile p dans la file s. La sortie s vaut **a b c d e * f + - * +**.

On suppose que vous disposez d'une **fonction priorite(t : Terme (E)) : Entier** qui donne pour chaque terme sa priorité avec les valeurs suivantes :

- priorite ('(') = 3 et priorite(')') = 3
- priorite('*') = 2 et priorite('/') = 2
- priorite('+') = 1 et priorite('-') = 1

De même, vous avez les méthodes suivantes pour déterminer le type d'un terme :

- **fonction estOperande(t : Terme (E)) : Booleen**
- **fonction estParentheseOuvrante(t : Terme (E)) : Booleen**
- **fonction estParentheseFermante(t : Terme (E)) : Booleen**
- **fonction estOperateur(t : Terme (E)) : Booleen**

On vous demande d'écrire la fonction qui reçoit l'expression infixée sous la forme d'un flot et qui retourne l'expression postfixée sous la forme d'une file.

3. Récursivité

On appelle partition d'un entier n un multi ensemble d'entiers strictement positifs dont la somme vaut n. Par exemple, 5=5 et 5=2+2+1 sont des partitions de l'entier 5.

Le but de cet exercice est de dénombrer les différentes manières de partitionner un entier n. On notera ce nombre p(n). On définit une fonction auxiliaire p(n,k) qui compte le nombre de partitions de l'entier n en exactement k parts.

1) Donner toutes les manières de partitionner chacun des entiers de 1 à 6.
2) Que vaut p(n,1) ?
3) Que vaut p(n,n) ?
4) Que vaut p(n,k) si n<k ?
5) Démontrer que dans le cas général p(n,k)=p(n-1,k-1)+p(n-k,k).
6) Ecrire une fonction qui calcule p(n).

4. Problème

Une fraction égyptienne est une somme de fractions unitaires, c'est-à-dire de fractions qui ont des numérateurs égaux à un et des dénominateurs entiers positifs, avec ces dénominateurs distincts deux à deux. Il peut être montré que tous les nombres rationnels positifs peuvent être écrits sous cette forme et ce, d'une infinité de façons différentes.
En effet, il est trivial d'exprimer toutes fractions par une somme de fractions unitaires en se permettant de répéter les termes comme dans l'exemple : ⅔ = ⅓ + ⅓
Mais si l'on exige que tous les dénominateurs soient distincts, à l'instar des Égyptiens durant l'Antiquité, cette représentation est toujours possible grâce à l'identité remarquable suivante :

$$1/a = 1/(a+1) + 1/(a(a+1))$$

Ainsi, en reprenant l'exemple ci-dessus : ⅓=1/(5+1)+1/(5(5+1)) d'où ⅔ = ⅓ + ⅙ + 1/30 . En appliquant le même procédé à chacune des fractions unitaires, 2/5 peut donc s'exprimer comme une multitude de fractions égyptiennes.

Soit la structure Fraction qui contient deux attributs entiers num et denom correspondant respectivement au numérateur et au dénominateur.
Ecrire la procédure **formeEgyptienne(f : Fraction, liste : Liste de Fraction (S))** qui prend une fraction f, dont le

numérateur et dénominateur sont strictement supérieurs à 1, et qui construit une liste des fractions d'une forme égyptienne. A vous de découper au mieux le problème en un ensemble de méthodes.

Vous avez à disposition les méthodes suivantes :

- **fonction recupererI(liste : Liste d'Element; i : Entier) : Element**. Cette fonction permet de récupérer le i-ème élément d'une liste;
- **procédure supprimerI(liste : Liste d'Element (E/S); i : Entier**). Cette fonction permet de supprimer le i-ème élément d'une liste.